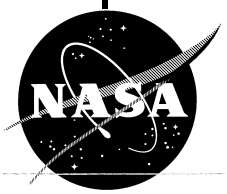
MISSION OPERATIONS AND DATA SYSTEMS DIRECTORATE

Control Center Systems Portable Spacecraft Simulator Proposal

June 1996



National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland

Control Center Systems Portable Spacecraft Simulator Proposal

June 1996

Prepared Under Contract NAS5-31500

Prepared by:

Matthew McCoy Date
Computer Sciences Corporation

Approved by:

Elizabeth L. Chandler Date
NASA Code 515.1

Goddard Space Flight Center
Greenbelt, Maryland

Preface

This document is a proposal to the Vision 2000 Control Center Systems (CCS) project for a CCS Portable Spacecraft Simulator (PSS). This proposal outlines the development of a CCS-PSS based upon an augmentation of the current Hubble Space Telescope (HST) PSS.

Questions concerning this document should be addressed to:

Elizabeth L. Chandler, Test Systems Branch
Code 515.1
Goddard Space Flight Center
Greenbelt, Maryland 20771

1. Introduction

This is a proposal for an augmentation of the Hubble Space Telescope (HST) Portable Spacecraft Simulator (PSS) to become a test tool for the Vision 2000 Control Center Systems (CCS) project. Vision 2000 documentation addresses the need for automated test tools. Also, the dependence on too many simulators having limited functionality despite high fidelity capabilities has created the desire for an "ideal test system". This proposal will address the current capabilities and configuration of the PSS, the goals of augmenting the PSS to be the CCS-PSS, and possible enhancements and modifications necessary to meet these goals.

1.1 Current PSS Specifications

The HST PSS was first developed on a Data General MSE14 platform and was used extensively in the development stages of the HST Ground System. The PSS was used for interface testing, launch site testing, and simulations for HST deploy and first servicing missions. After the first servicing mission, it was determined that the PSS needed to be ported to the current VME platform for the second servicing mission. The VME hardware and C programming language makes the PSS a good candidate for UNIX based CCS-PSS development.

The current capabilities of the HST PSS can be categorized as low fidelity. However, the PSS has many features that compensate for not being database driven. The PSS is the only HST related simulator that can output all telemetry formats (Engineering, DF224, and Science). The PSS has the ability to modify an abundant number of bytes in an engineering format and download any DF224 or NSSC-I loads. Since the PSS is portable, it can be operated anywhere as long as the correct connections are present, namely a NASCOM interface.

The HST PSS is designed as a test tool. The CCS-PSS will be a test and training tool. The command logging and delogging functions allow the user to analyze a sequence of commands or a command load broken down to the op codes and function codes and printed in the DM01 format. All instrument packet headers are simulated and all science packet types are transmitted. Solid State Recorder (SSR)/Science Tape Recorder (STR)/Engineering Tape Recorder (ETR) playbacks are easily built using the Engineering to DRAM function and the Science to DRAM function. These functions are already very useful for testing of SSR capture at the STOCC. DF224 and NSSC-I loads and dumps are performed exactly since all memory areas are simulated. With the creative use of scenario files, many functions can be combined and time activated sequentially. The PSS has the ability to transmit software generated 'canned' telemetry or a telemetry file

stored on the VME hard disk or 8mm tape. Telemetry files can be real spacecraft data from any project source such as PACOR or DCF, or HST Simulator data.

1.1.1 Hardware Description

The HST PSS platform (Figure 1-1) consists of a collection of 6U cards which occupy single slots within a VME chassis. The simulator platform is composed of: a Motorola MVME 167-004 card (hosting a MC68060 microprocessor and an Ethernet controller); four AVTEC High-Speed Input/Output (HSIO) cards (three are active, one is a spare); a 128MB DRAM (an upgrade to a 256 Mbyte DRAM card is planned for by 8/30/95); a 1 GB Hard Disk Drive; a 3.5" 1.44MB Floppy Disk Drive; and an 8mm Cartridge Tape Unit. The MVME 167 utilizes the PDOS operating system. The user interface program (FE.EXE) executes under DOS, Windows, Windows NT, or OS/2 on a PC or PC compatible laptop with an RS-232 port.

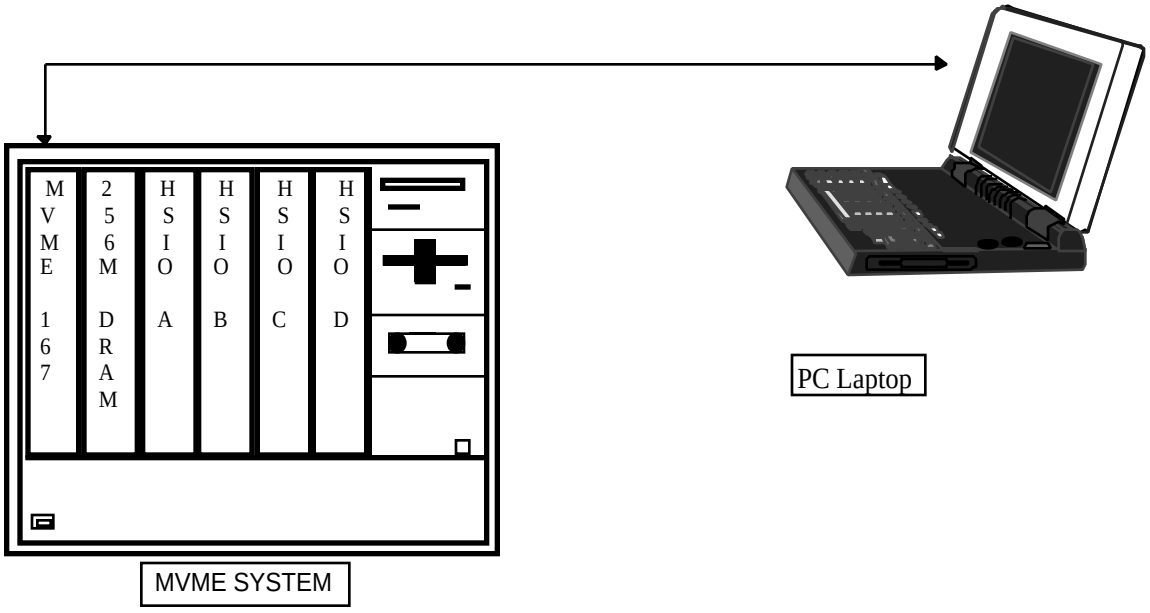


Figure 1-1. HST PSS Platform with PC laptop

1.1.2 Software Description

The PSS is written primarily in the C programming language on the PDOS operating system. There is extensive use of the SOC PSS baseline library functions. The communications with the user interface and the HSIO interface is common among all PSSs. There are a few MC68000 assembly language routines related to the PN encoding of the science data and the software sync search for the command logging. There are several PDOS system calls that are used in each task. The number of system calls should average approximately three for each task. It is possible to port a majority of this code to another platform, namely the UNIX based platform planned for the CCS-PSS.

2. CCS-PSS Goals

Vision 2000 documentation details the attributes for an “ideal test system” (Table 1-1). Most of these ideal test system capabilities are obtainable by augmenting the PSS to be the CCS-PSS with the following goals: 1.) Increase the fidelity of the PSS by becoming database driven and add triggered stored response capabilities; 2.) Increase the automation of the simulations by a system that is active 24 hours a day and configured easily by saving/restoring test scenarios and configurations; 3.) Decrease operator interface dependability by telemetry generation via database and command responses and verification via database; 4.) Use legacy software to build UNIX based CCS-PSS; 5.) Incorporate Java utilizing Netscape Navigator 2 for the Graphical User Interface (GUI); 6.) Allow multiple users on CCS workstation via Internet Protocol LAN. With these goals in mind, the CCS-PSS can satisfy many of the high level requirements, for the “ideal test system”. These requirements are shaded in Table 2-2.

The main goal to achieve is compatibility with the CCS. The CCS-PSS will be integrated with the CCS. This can be accomplished by sharing a common user interface. The CCS-PSS will be controlled from a CCS workstation. The CCS-PSS will contain the following aspects:

- Workstation: HP, SUN, SGI on CCS LAN
- Legacy Software/C or C++
- NASCOM Interface/IP
- UDP/IP interface
- UNIX/C++/Java user interface
- Simulator cloned “N” number of times
- Remote login via X-terminal/ portable to remote locations
- Tight integration with the CCS test automation tools and CCS user interface
- CCS-PSS displays via Java GUI

There will be a long term reduction in maintenance costs by a conversion of the PSS software to run on a UNIX based platform and maintained in a UNIX/C++/Java environment.

REF#	ATTRIBUTE
1	Flexibility.
2	Configuration under control of tester (which hardware and software components to include in the test).
3	Control of the operations environment (simple setup).
4	Constraints handled transparently.
5	Resources always available, even for concurrent testing.
6	Use automation for repetitive tasks.
7	Simple data analysis and interpretation.
8	Rapid distribution of test results.
9	Minimize maintenance.
10	Minimize expertise required from testers.
11	Easy simulation for stress/exception testing.
12	Self-documenting tests.
13	Minimize manual analysis (manual comparison, verification).
14	Good (computer-aided?) coordination.

Table 2-1. Attributes for CCS "ideal test system"

REQ#	REQUIREMENT	ATTR.
1	Log significant events, configuration, data, under control of tester	1, 2, 3
2	Test tools should be modular and easily distributed (COTS tools, ground system, simulators, etc.)	1, 2, 3
3	Support testing at remote sites	1, 2, 3
4	Selectable portions of simulators	1, 2, 3
5	Selectable versions of subsystems	1, 2, 3
6	Selectable environment and databases	1, 2, 3
7	Configuration, ops. constraints automatically handled (i.e., non value-added components automatically configured out (with operator override))	4, 6, 10
8	Operator can override constraints and introduce error conditions	4
9	Redundant capabilities with sufficient resources to eliminate scheduling conflicts	5
10	"Canned" baseline operational configurations, even for non-nominal situations	5, 6, 10
11	Synthesis of test results; extraction of key parameters; automated differencing; some diagnostics	6, 7, 13
12	User-defined level of reports, logs, event notification	7
13	Real-time visibility into test conditions, problems, test success/failure	7, 10
14	Data analysis tools	6, 7, 13
15	Access to test schedules, plans, etc. for all who need to know	8
16	Notification of "test flashes" to potentially affected parties (Test Bulletin Board with digested information)	8
17	User-controlled simulation range, responses	11
18	Spacecraft simulation which is modular and user-configurable	11
19	CM-specified report generation and archive	12
20	User help and access to retrieve test results	6, 12
21	Automated coordination of activities, plans, schedules, resources	6, 12, 14

Table 2-2. High-level Requirements for an 'ideal test system'

3. CCS-PSS Description

An augmentation of the PSS to the CCS-PSS is divided into three sections: 1.) the hardware description of a proposed system; 2.) modifications of functions resident on the VME 167 board; 3.) an enhancement of the user interface to be compatible with CCS.

3.1 Hardware Description

The hardware platform (Figure 3-1) will consist of a collection of 6U cards which occupy single slots within a VME chassis. The CCS-PSS will have a front-end user interface component and a back-end VME simulator component. The front-end system will provide the user interface, project database interface, and other support software. The VME component hosts the spacecraft simulator functions. The CCS-PSS front end component will consist of a Web-based or X-based GUI running under UNIX or Windows on a workstation or PC networked with an external component contained in the simulator platform chassis. This external component will consist of a Motorola RISC single board VME-based computer (an MVME 187-003B card hosting a MC88100 RISC processor running at 25 MHz with 16 MB memory), TVME712-E transition module/Ethernet transceiver, 1 GB hard drive, and streamer tape drive. The single board computer supports the UNIX System V operating system. A Pentium-based notebook PC running at 90 MHz with 16 MB of memory, a 520 MB removable hard drive, a 640x840 active matrix color display, serial and parallel ports, a mouse pointing device, and an Ethernet (thin net) card is planned for portability for use at remote sites.

The simulator component will be composed of: a Motorola MVME 167-004 card (hosting a MC68060 microprocessor and an Ethernet controller); three AVTEC High-Speed Input/Output (HSIO) cards; one PC Synchronizable Time Generator card; a 384 MB DRAM; a 1 GB Hard Disk Drive; a 3.5” 1.44MB Floppy Disk Drive; a 4-mm Cartridge Tape Unit; and, a bubble jet printer. The MVME 167 runs under the PDOS operating system. Upgrading the 167 card to the MVME 187-003B card is also a likely scenario under consideration.

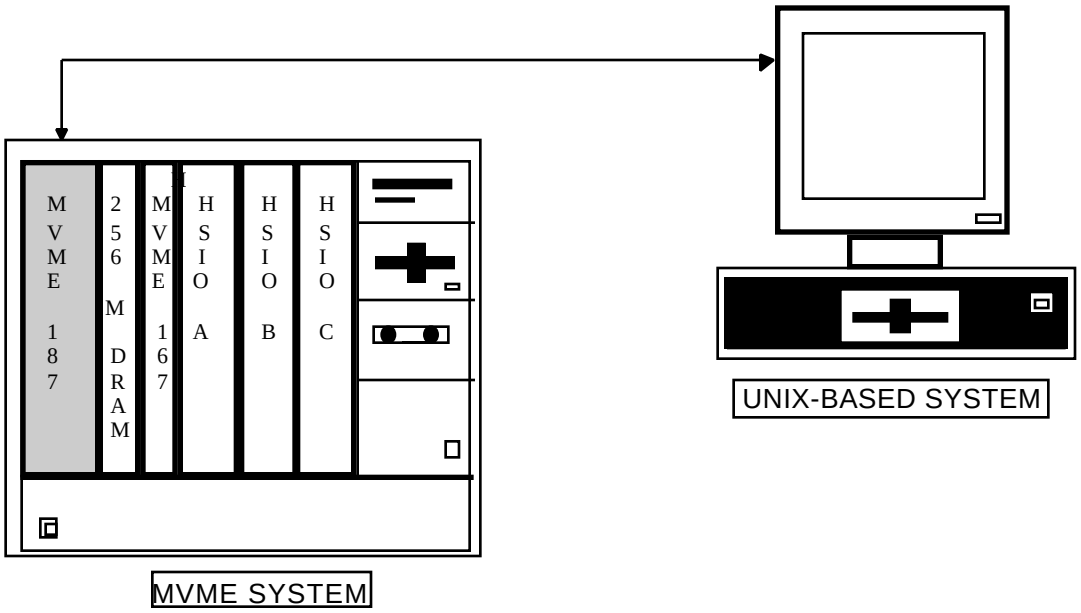


Figure 3-1. CCS-PSS Platform with X-terminal

3.2 Software Description

The CCS-PSS will be divided into three software configuration items (SWCIs), which include the CCS-PSS User Interface SWCI, CCS-PSS Utilities SWCI, and Spacecraft Simulation SWCI (see Figure 3-1). The Spacecraft Simulation SWCI is further divided into five subsystems, which include the Initialization Subsystem, Configuration Support Subsystem, Telecommand Subsystem, Telemetry Subsystem, and Database Driven Triggered Stored Response Subsystem (in lieu of modeling). The five subsystems of the Spacecraft Simulation SWCI run on the VME main processor, and the remaining two SWCIs will run on the CCS-PSS front end.

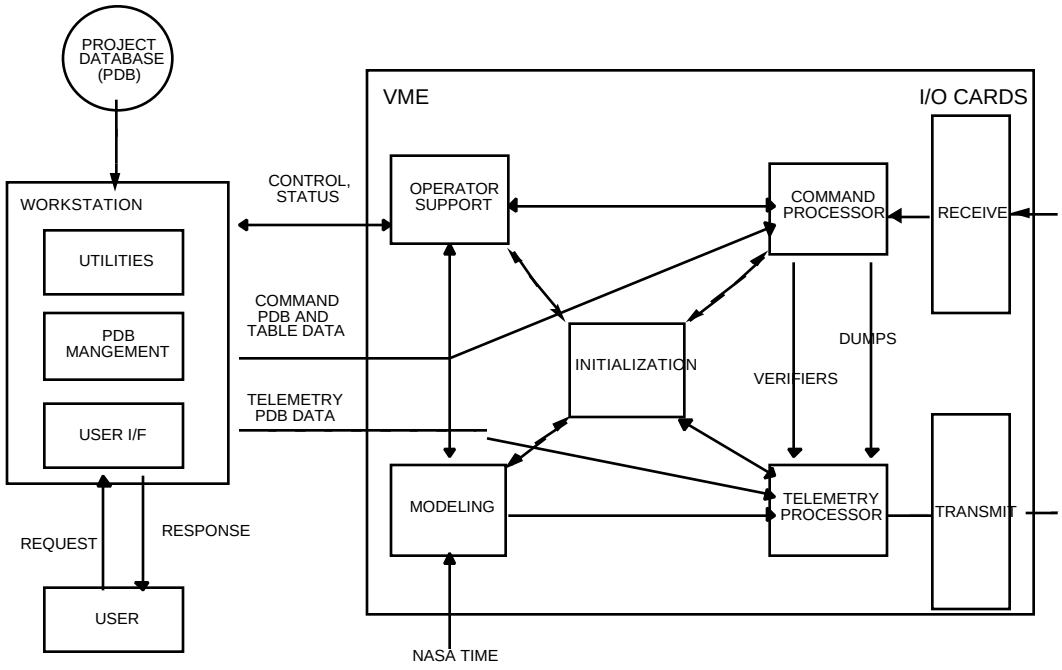


Figure 3-2. CCS-PSS Software Architecture

3.2.1 VME 167 Function Modifications

An enhancement for VME 167 related capabilities should consist of the following:

- Telemetry generation via database for engineering formats
- Command execution and verification via telemetry verifiers (database retrieved)
- The use of command triggered scenario files to allow triggered stored responses in place of spacecraft models
- Modification and display of telemetry mnemonics in engineering units
- Inclusion of real Reed-Solomon segments in the science data. This could be performed via hardware or software.
- Hardware modification to at least 384 Mbytes of DRAM

With this augmentation, the PSS would be a very useful test tool for the CCS developers if the PSS was made available at the CCS development facility. With the addition of the interface to the project database, the PSS would be functionally equivalent (for commands and telemetry) to the TPOCC Advanced Spacecraft Simulator (TASS) used as the primary test tool for projects using TPOCC as it's control center. The DRAM increase

would provide Engineering and Science SSR or STR/ETR dumps at durations typical of the standard TDRSS event.

3.2.2 CCS-PSS User Interface Description

The CCS-PSS user interface shall be compatible with CCS. The CCS-PSS shall incorporate the use of Web-based technologies, specifically the Java programming language. This would allow the CCS-PSS user interface to be completely compatible with CCS, i.e. CCS-PSS users will be able to monitor and control the simulator from a CCS workstation.

Since inexpensive Java-capable Web browsers are available on all modern platforms, simulator users will be able to login and monitor displays from remote locations. Web-based applications can be run without modification on either an internal, high-speed network (known as an "Intranet") or the Internet. Basing the user interface on the Web provides flexibility in selecting from a broad range of products (e.g., hypertext documentation, collaboration tools, database access) available from a variety of vendors instead of being locked into proprietary or single-platform solutions.

Java is composed of the following components:

- An object-oriented (OO) programming language specification based on C++ and similar to Smalltalk.
- A Virtual Machine (VM) specification that allows Java executables to run on any platform.
- A class library which implements a large number of useful objects.
- A development environment consisting of a compiler, debugger, interpreter, applet viewer, and documentation generator.
- A Web browser with an embedded Java VM (e.g., Netscape 2.0).

MVME 187 communication software will also be developed using the HP. These tasks will be written in the C programming language and integrated within the CCS-PSS User Interface and Utilities SWCI. The communication tasks will be developed for both the MVME 187/Unix card and the MVME 167/PDOS card. There will be two TCP/IP socket connections between the cards. One socket will connect the MVME 187 based client to the MVME 167-based server and the other will connect the MVME 167 based client to the MVME 187-based server. This system will be tested using a GUI and spacecraft simulation prototype which will verify communication between the elements.

It should be noted that several hardware configuration options are still under investigation. An upgrade from the MVME 167/PDOS system to the MVME 187/UNIX system would enhance the interface between the simulator and GUI

components. The addition of a second cpu board for the simulator component is being considered for the soft requirement that the CCS-PSS to be cloned “N” number of times.

4. Benefits of PSS Augmentation

The HST PSS currently is a valuable test tool for the intense preparation of the upcoming servicing mission. However, testing various functions of a ground system require a simulator or test tool with much more flexibility, fidelity, and availability. The proposed CCS-PSS system would provide the CCS with a test tool that could be used by developers for all levels of testing described in the CCS Master Test Plan: unit testing, integration testing, application testing, and thread testing. Each test phase: development and integration, shadow-mode, and deployment phases would benefit from a simulator with the enhanced functions of the CCS-PSS. Some of the CCS documentation has even suggested this idea. Of the five functions described in the CCS Master Test Plan, the front end processing function, the command processing function, and the data management function would benefit the most from CCS-PSS.

The proposed CCS-PSS would be categorized as a medium fidelity simulator. From past experience with many GSFC related projects, it has been found that a medium fidelity is all that was required to test a ground system through its development phase. A high fidelity simulator is much more useful for FOT training than ground system testing. Since the CCS-PSS will have the capability to transmit pre-recorded spacecraft data, this function would supplement the fidelity of the software generated telemetry.

4.1 CCS Thread Testing

The following table (Table 4-1) is derived from the Level-2 CCS Thread Definitions table. The entries are the level-2 threads that have been identified as directly testable via the CCS-PSS. There are some level-2 threads that are indirectly testable, but those have not been identified due to lack of thread descriptions. The four digit sub-segment identifiers are not included in this table.

16.1.1	**HST Downlink)**	**FEP**	**(16.1.2)**	**19**	**Change in communications status**
16.1.2	**FE Event**	**SYM**	**(16.1.3)**		**Notify monitoring of change in communications**
17.1.1	**HST Downlink**	**FEP**	**(17.1.2, 17.1.3, 17.1.4)**	**20**	**Downlink of Real-Time Telemetry Data**
17.1.2	**Processed Telem**	**SYM**	**(17.1.5, 17.1.7)**		**Perform real-time monitoring**
17.1.3	**FE Event**	**SYM**	**(17.1.7)**		**Log receipt of real-time telemetry data**
17.1.4	**FE Data**	**DMG**	**End**		**Store real-time telemetry for user access**
17.1.5	**Command Directive**	**CMD**	**(17.1.6)**		**Command directive issued as a result of real-time anomaly detection**
18.1.1	**HST Downlink**	**FEP**	**(18.1.2, 18.1.3, 18.1.4)**	**21**	**Dump Recorded Telemetry Information**
18.1.2	**FE Event**	**SYM**	**End**		**Log receipt of recorded telemetry data**
18.1.3	**FE Data**	**DMG**	**(18.1.4, 18.1.5)**		**Store recorded telemetry data**
19.1.1	**HST Downlink**	**FEP**	**(19.1.2, 19.1.3, 19.1.4)**	**22**	**Playback Data from WSC/DSN**
19.1.2	**FE Event**	**SYM**	**End**		**Log Playback Data Arrival**
19.1.3	**FE Data**	**DMG**	**(19.1.4)**		**Initiate merge of playback data, as necessary**
19.1.4	**DM Event**	**SYM**	**End**		**Playback Data Arrival**
21.1.1	**ICS Datastore**	**CMD**	**(21.1.2)**	**28**	**Time to uplink a command**
21.1.2	**State Data Req**	**SYM**	**(21.1.3)**		**Request validation of specific aspects of spacecraft state**
21.1.3	**State Data Resp**	**CMD**	**(21.1.4, 21.1.5, 21.1.13)**		**Indication of current spacecraft state based on real-time monitoring data**
21.1.4	**State Change Info**	**SYM**	**End**		**Prepare for transmission of a specific spacecraft command**
21.1.5	**Command Load**	**FEP**	**(21.1.6, 21.1.7)**		**Format command loads for transmission to the spacecraft**
21.1.6	**HST Uplink**	**WSC/DSN/JSC**	**(21.1.7, 21.1.8)**		**Uplink commands to the spacecraft**
21.1.7	**FE Event**	**SYM**	**End**		**Record uplink of command load**
21.1.8	**Transmission Status**	**CMD**	**(21.1.9, 21.1.10, 21.1.11)**		**Notify Commanding of the status of the command load uplink**
21.1.9	**State Change Info**	**SYM**	**(21.1.11)**		**Notify Monitor to expect successful command execution**
21.1.10	**CMD Events**	**SYM**	**End**		**Record command events into the system event log**
21.1.11	**System Events**	**CCM**	**(21.1.12)**		**User assistance required in dealing with detected problem**

21.1.12	**Collaboration Notice**	**CCSU**	**End**		**Request user intervention**
22.1.8	**Transmission Status**	**CMD**	**(22.1.9, 22.1.10, 22.1.11)**		**Notify Commanding of the status of the command load uplink**
22.1.9	**State Change Info**	**SYM**	**(22.1.10)**		**Notify Monitor to expect successful command execution**
22.1.10	**CMD Event**	**SYM**	**(22.1.11)**		**Log status of command execution**
22.1.11	**System Events**	**CCM**	**(22.1.12)**		**User assistance required in dealing with detected problem**
23.1.1	**Analysis Request**	**SYM**	**(23.1.2)**	**30**	**Automatically started analysis of engineering data**

Table 4-1. Level-2 Thread Definititions Testable via CCS-PSS

6. Other SOC Resources

6.1 Network Utilities

Code 515 is proposing in one of the options (see Section 8; Option 4) to deliver Network Control Center(NCC) Message Simulator(MESSIM), Johnson Spaceflight Center(JSC) Simulator(JSCSIM), JSC Mission Control Center Simulator(MCCSIM) capabilities to a PC based platform. This platform would be resident inside the CCS and used in conjunction with the CCS-PSS for testing and network simulations.

6.2 PSS Test Tools

The tools used by the SOC to test the current PSS will be used by the CCS-PSS developers and acceptance test team. These tools include the Data Quality Monitor (DQM), Command Generator (CGEN), HST Telemetry Check Utility (HSTCHECK), and the HST POCC Emulator (HPOCC). HPOCC is still under development.

7. Spacecraft Modeling

One aspect of an augmentation of the HST PSS to the CCS-PSS that needs to be addressed at a high level is the addition of spacecraft modeling. Many of the PSSs built at the Simulations Operations Center (SOC) contain modeling functions at various levels of fidelity. Most models are for analog telemetry mnemonics such as currents, voltages, temperatures, and pressures that are ramped/scripted using simple functions that can be activated/deactivated via command execution. The HST Simulator has dynamic, medium fidelity models that are HST specific for the instruments as well as Support Systems Module(SSM) subsystem components.

Several re-engineering efforts have been performed on the HST Simulator, many of them in C. However, the HST Simulator is primarily a FORTRAN based system. Derived from discussions with the HST Simulator task leader, it has been determined that integrating spacecraft models from the HST Simulator into the CCS-PSS would require some initial research to determine which models could be integrated, and the conversion of software from FORTRAN to C where appropriate. This would require knowledgeable personnel from the PSS and from the HST Simulator.

The spacecraft models would cover a wide range of spacecraft elements. The following SSM subsystems would be included:

- Thermal Control Subsystem (TCS)
- Pointing Control Subsystem (PCS)
- Electrical Power Subsystem (EPS)
- Safing Subsystem
- Instrumentation and Communications Subsystem (I&CS)
- Data Management Subsystem (DMS)

8. HST PSS Proposal Options

Augmenting the HST PSS to the CCS-PSS should be a multi-step process. Table 8-1. presents a list of the high level capabilities of the current PSS with comparisons to capabilities divided into five options. The current PSS capabilities represent the delivered PSS system, Release 4.0, as of May 1, 1996. The five proposal options are:

- Option 1 - Current PSS Capabilities with Blocked Mode
- Option 2 - PSS with Project Database Interface
- Option 3 - CCS-PSS; UNIX based, CCS user interface
- Option 4 - Network Utilities
- Option 5 - Inclusion of HST Simulator Models

PSS CAPABILITIES COMPARISON TABLE

CURRENT PSS CAPABILITIES	OPTION 1 - CURRENT PSS CAPABILITIES WITH BLOCKED MODE
The PSS shall display command bits to console in hex and DM01 formats.	The PSS shall display command bits to console in hex and DM01 formats.
The PSS shall receive commands in serial mode.	The PSS shall receive commands In blocked or serial mode.
Command receipt time shall be displayed at the msec level	Command receipt time shall be displayed at the msec level
The appropriate command counters shall be updated upon command validation and execution.	The appropriate command counters shall be updated upon command validation and execution.
DF-224 and NSSC-I checksums shall be performed on all loads.	DF-224 and NSSC-1 checksums shall be performed on all loads.
	Aft Flight Deck Commands will be executed with appropriate command verifier values.
The PSS shall log and delog all incoming commands. Command blocks shall be logged with a time tag with an lsb of 10 msec.	The PSS shall log and delog all incoming commands. Command blocks shall be logged with a time tag with an lsb of 10 msec.
The command delog file shall consist of command bits in hex, in DM01 format, command block time tags, and the capability to compute time difference between blocks.	The command delog file shall consist of command bits in hex, in DM01 format, command block time tags, and the capability to compute time difference between blocks.
The PSS shall be able to transmit primarily static engineering data at the 500 bps, 4 kbps, and 32 kbps rates for all engineering formats.	The PSS shall be able to transmit primarily static engineering data at the 500 bps, 4 kbps, and 32 kbps rates for all engineering formats.
The engineering data shall consist of minor frames where the minor frame words will be initialized to the word number; the dynamic portion of the data will consist of command counters, major/minor frame counters, specified initial values and scripted mnemonics.	The engineering data shall consist of minor frames where the minor frame words will be initialized to the word number; the dynamic portion of the data will consist of command counters, major/minor frame counters, specified initial values and scripted mnemonics.
The specified initial values and scripted values shall consist of up to 75 telemetry mnemonics, their telemetry locations, initial value, maximum and minimum values, time between change (should be divisible by 10 msec), type of ramping function (step, wave, sawtooth, etc.), and step value. This information shall be stored in ASCII data files and read in during PSS initialization.	The specified initial values and scripted values shall consist of up to 150 telemetry mnemonics, their telemetry locations, initial value, maximum and minimum values, time between change (should be divisible by 10 msec), type of ramping function (step, wave, sawtooth, etc.), and step value. This information shall be stored in ASCII data files and read in during PSS initialization.
The PSS shall be able to transmit pre-recorded engineering data from 8 mm tape or from the hard disk.	The PSS shall be able to transmit pre-recorded engineering data from 8 mm tape or from the hard disk.
The PSS shall have the capability to dump, in real-time, the contents of a minor frame to the console.	The PSS shall have the capability to dump, in real-time, the contents of a minor frame to the console.
The PSS shall be capable of transmitting the engineering data in serial mode.	The PSS shall be capable of transmitting the engineering data in serial or blocked mode.
The PSS shall be able to record engineering telemetry to DRAM card to emulate engineering tape recorder or engineering solid state recorder playback.	The PSS shall be able to record engineering telemetry to engineering section of DRAM card to emulate engineering tape recorder or engineering solid state recorder playback.

The PSS shall have the capability to manipulate data patterns in the engineering data via operator type-in. The location(s) and range will be on a word (8 bit) boundary.	The PSS shall have the capability to manipulate data patterns in the engineering data via operator type-in. The location(s) and range will be on a word (8 bit) boundary.
The PSS shall be able to transmit 'canned' real-time science at 4 kbps or 1.024 Mbps.	The PSS shall be able to transmit 'canned' real-time science at 4 kbps or 1.024 Mbps.
The canned science data shall consist of static data and the appropriate packet headers for instrument packets, SHPs, UDLs, and status buffer packets.	The canned science data shall consist of static data and the appropriate packet headers for instrument packets, SHPs, UDLs, and status buffer packets.
The canned science data shall be PN encoded and/or RS encoded if selected. The RS encoding shall consist of a static segment that matches 14 segments of fill data with no spacecraft time.	The canned science data shall be PN encoded and/or RS encoded if selected. The RS encoding shall consist of a static segment that matches 14 segments of fill data with no spacecraft time.
The PSS shall be capable of transmitting real-time or playback science data from an 8mm tape.	The PSS shall be capable of transmitting real-time or playback science data from an 8mm tape.
The PSS shall have the capability to manipulate data patterns in the science data via operator type-in.	The PSS shall have the capability to manipulate data patterns in the science data via operator type-in.
The PSS shall have the capability of transmitting pre-recorded science data from the DRAM to simulate science tape recorder or science solid state recorder playbacks	The PSS shall have the capability of transmitting pre-recorded science data from the DRAM to simulate science tape recorder or science solid state recorder playbacks
The PSS shall accept and internally store DF-224/NSSC-1 memory loads.	The PSS shall accept and internally store DF-224/NSSC-1 memory loads.
The PSS shall have the capability to manipulate the data contents of memory areas in real-time via operator type-in.	The PSS shall have the capability to manipulate the data contents of memory areas in real-time via operator type-in.
PSS shall have the capability to display the contents of any operator selected memory area.	PSS shall have the capability to display the contents of any operator selected memory area.
The PSS shall have the capability to transmit DF-224/Co-processor and NSSC-I dump formats.	The PSS shall have the capability to transmit DF-224/Co-processor and NSSC-I dump formats.
The PSS will initialize the memory area data with the address or word number, i.e. NSSC-I location 200 will contain '200'	The PSS will initialize the memory area data with the address or word number, i.e. NSSC-I location 200 will contain '200'
The PSS shall use the most recent versions of the generic PSS user interface files, i.e. FE.EXE and FGEN.EXE. This will include the standard capabilities for displays, type-ins, and session logs.	The PSS shall use the most recent versions of the generic PSS user interface files, i.e. FE.EXE and FGEN.EXE. This will include the standard capabilities for displays, type-ins, and session logs.
The PSS shall provide the operator with the status of program and spacecraft parameters for command and telemetry subsystems.	The PSS shall provide the operator with the status of TBD program and spacecraft parameters.

	OPTION 2 - PSS WITH PROJECT DATABASE INTERFACE
The appropriate command counters shall be updated upon command validation and execution.	Command verification shall include setting of telemetry verifiers defined in DM01. Command verifiers shall be retrieved from project database files.
The appropriate command counters shall be updated upon command validation and execution.	Command execution shall include responses that perform telemetry related functions such as selection of telemetry formats and start/stop telemetry transmission.
The appropriate command counters shall be updated upon command validation and execution.	Command execution shall include the use of triggered stored responses defined in database files that connect commands to scenario files.
The command delog file shall consist of command bits in hex, in DM01 format, command block time tags, and the capability to compute time difference between blocks.	The command delog capability shall add a delog file comparison. The file comparison shall include notification of differences in command sequences and significant (TBD) time differences between command blocks.
The engineering data shall consist of minor frames where the minor frame words will be initialized to the word number; the dynamic portion of the data will consist of command counters, major/minor frame counters, specified initial values and scripted mnemonics.	Engineering telemetry shall consist of current values of all telemetry mnemonics with locations defined by project database files.
	The current value for the telemetry mnemonics shall be displayed in engineering units as well as PCM counts. Telemetry engineering conversion algorithms shall be retrieved from project database files.
	The PSS shall provide the capability to display the telemetry location(s) for any telemetry mnemonic defined by the project database files.
	The PSS shall have the capability to set initial values for any telemetry mnemonic defined in the project database.
	The PSS shall emulate the solid state recorder subsystem.
	The PSS shall use a PTP to interface with CCS via LAN.

	OPTION 3 - CCS-PSS; CCS USER INTERFACE
	The CCS-PSS user interface shall consists of a Web-based or X-based GUI running under UNIX or Windows on a workstation or PC.
	The CCS-PSS user interface shall be compatible with the CCS user interface.
	The CCS-PSS shall have remote terminal login via Netscape.
	The CCS-PSS shall use the Java Programming Language, using common components with the CCS.
	The CCS-PSS shall have the capability to display real-time simulator test data.
	The CCS-PSS shall provide on-line access to project reference data.
	The CCS-PSS user interface shall provide the capability to perform test scripting via scenario files for test automation.
	The CCS-PSS shall be ported to a UNIX based platform
Note: current PSS has the capability to transmit telemetry from the DRAM card at 5 Mbps.	The CCS-PSS shall provide the capability for future instruments and higher data rates up to 5 Mbps.
	The CCS-PSS shall provide the capability to query data archives.

	OPTION 4 - NETWORK UTILITIES
	The CCS-PSS shall provide the capability to perform the following functions resident in the Johnson Spaceflight Center (JSC) Simulator (JSCSIM) utility:
	The CCS-PSS shall provide the capability to process real-time and playback orbiter telemetry at 64, 96, 128, and 192 kbps.
	The CCS-PSS shall provide the capability to generate 32 and 72 kbps uplink in blocked or serial mode.
	The CCS-PSS shall provide the capability to build single stage and two-stage command sequences.
	The CCS-PSS shall provide the capability of command echo and command logging.
	The CCS-PSS shall provide the capability to display site status messages.
	The CCS-PSS shall provide the capability to perform the following functions resident in the JSC Mission Control Center (MCC) Simulator (MCCSIM) utility:
	The CCS-PSS shall provide the capability to receive and process all orbiter Operational Downlink (OD) data.
	The CCS-PSS shall provide the capability to output one or two Payload Data Interleaver (PDI) streams and one Payload Parameter Frame (PPF) stream.
	The CCS-PSS shall provide the capability to receive and validate POCC command blocks and interleave valid payload commands into shuttle command stream.
	The CCS-PSS shall provide the capability to output Command Acceptance Pattern (CAP) blocks for single stage and two stage commands.
	The CCS-PSS shall provide the NCC message interface capability equivalent in the NCC Message Simulator (MESSIM):
	The CCS-PSS shall provide the capability to transmit User Performance Data (UPD) messages.
	The CCS-PSS shall provide the capability to validate and respond to Ground Control Messages (GCMR)s.
	The CCS-PSS shall provide the capability to validate and respond to NCC scheduling messages.
	The CCS-PSS shall provide the capability to validate and respond to communications test messages.

	OPTION 5 - HST SIMULATOR MODELS
	The CCS-PSS shall incorporate TBD HST Simulator models for the power subsystem.
	The CCS-PSS shall incorporate TBD HST Simulator models for the thermal subsystem.
	The CCS-PSS shall incorporate TBD HST Simulator models for the C&DH subsystem.
	The CCS-PSS shall incorporate TBD HST Simulator models for all spacecraft instruments.

Table 8-1. PSS Capabilities Comparison Table